sainho 02月06日

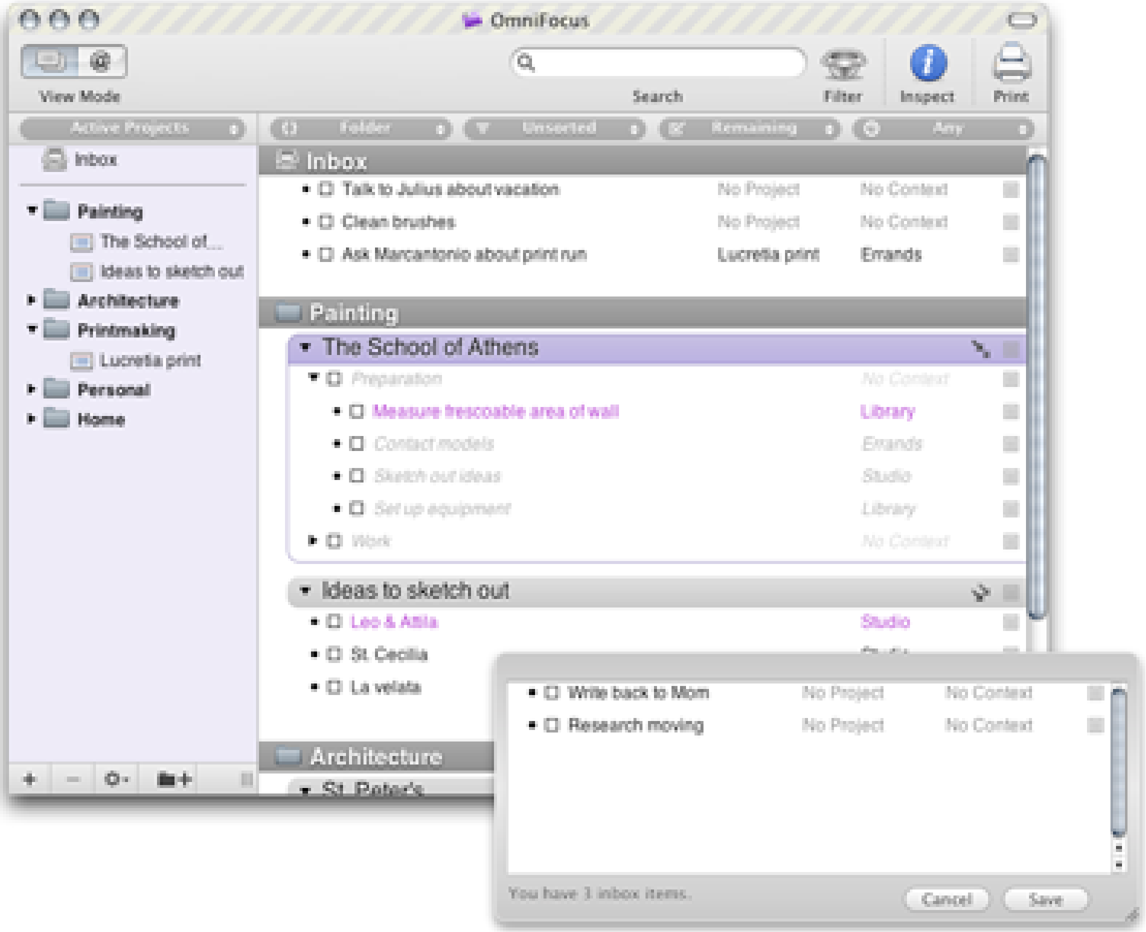
第六章 | 使用 AppleScript 拓展 OmniFocus

| 本文为付费栏目文章，您已订阅，可阅读全文 |

一提到和脚本、代码或者自动化相关的应用，很多人都会未战先怯。认为这类应用虽然功能强大，但学习曲线陡到难以逾越，从而敬而远之。但对于花了大价钱购买 OmniFocus for Mac 专业版的你来说：

如果你不使用 AppleScript，那就是对专业版 OmniFocus **最大的浪费。**

相比 2Do、Things 等任务管理软件，OmniFocus 最明显的优势，其实就在于它对 AppleScript 的深度支持。OmniGroup 从 1989 年创立之初，就开始为苹果系操作系统开发应用[1]。单看 OmniFocus，也有近 10 年的开发历史。


OmniFocus
View Mode
Search
Filter
Inspect
Print
Active Projects
Folder
Unsorted
Remaining
Any
Inbox
Painting
The School of...
Ideas to sketch out
Architecture
Printmaking
Lucretia print
Personal
Home
Talk to Julius about vacation
No Project
No Context
Clean brushes
Ask Marcantonio about print run
Errands
The School of Athens
Preparation
Measure frescoable area of wall
Library
Contact models
Sketch out ideas
Studio
Set up equipment
Work
Ideas to sketch out
Leo & Attila
St. Cecilia
La velata
St. Peter's
Write back to Mom
Research moving
You have 3 inbox items.
Cancel
Save

第一版 OmniFocus

所以打开 OmniFocus 的 AppleScript 字典一看，和 Things 孤零零的一套 Suite 相比，OmniFocus 靠着 Omni 系其他应用的协同优势，足足有 6 套 Suite。这意味着 OmniFocus 的 AppleScript 的写法能**更加灵活，功能也强大**。

短短一章很难说尽 AppleScript 在 OmniFocus 里的所有妙用。所以我们将从实际的应用场景出发，去看看有哪些切中痛点的 AppleScript。对于之前没有接触过 AppleScript 的读者，我将

尽可能地详细介绍这些脚本的使用方法，让大家可以直接上手使用，享受 AppleScript 的便利；而对于有经验的读者，我们将一探这些 AppleScript 的运行逻辑，去了解开发者的巧妙构思。

本文提及的脚本可以 点此下载。

AppleScript 是什么？

祛魅 AppleScript 的第一步，要从它的定义开始。在《AppleScript Language Guide》中如此定义 AppleScript：

> AppleScript is a **scripting language** created by Apple. It allows users to directly control **scriptable Macintosh applications**, as well as parts of macOS itself.

当我们抛开编程语言学家般极致严谨的视角，去阅读这句话时，会发现这段定义有两个重点词：「scripting language」 和「control scriptable Macintosh applications」。前者说明了 AppleScript 的身份，后者显示了它的特别之处。

AppleScript 是一门脚本语言（scripting language），[illegible]你没有学习过其他编程语言，可能会很难分清脚本语言和传统的 C 语言有什么异同。但如果单从「脚本」这个词出发，可能就好理解得多。

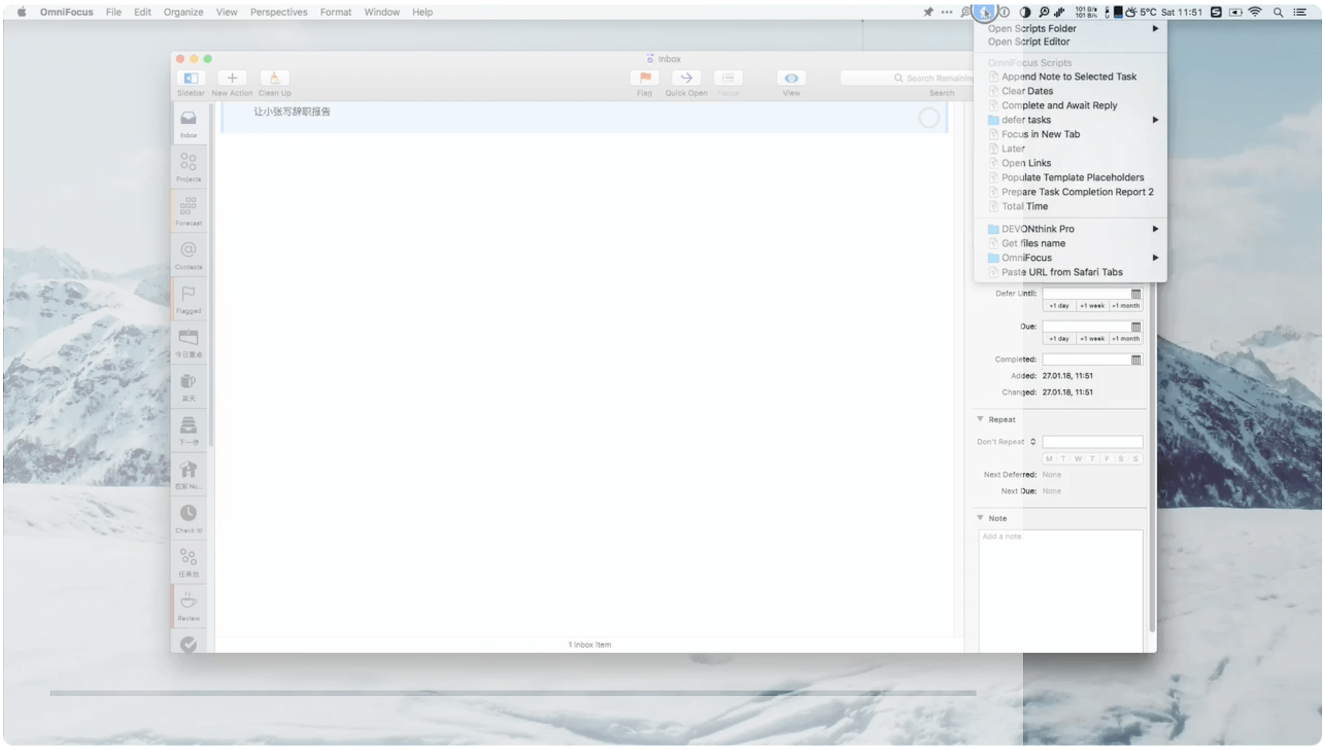
传统的戏剧舞台演出，都会有舞台工作者撰写的脚本，用来设计演员走位、台词和舞美等元素。表演时按照脚本执行，这样即使每场都是现场演出，也能保证观众能享受到水准一致的表演。而 AppleScript 也是这样，它并不是用来给程序开发新功能的语言，而是将程序现有的各个功能**进行编排，依次运行**，最终达到使用者想要的效果。

Complete and Await Reply（完成并等待回应）

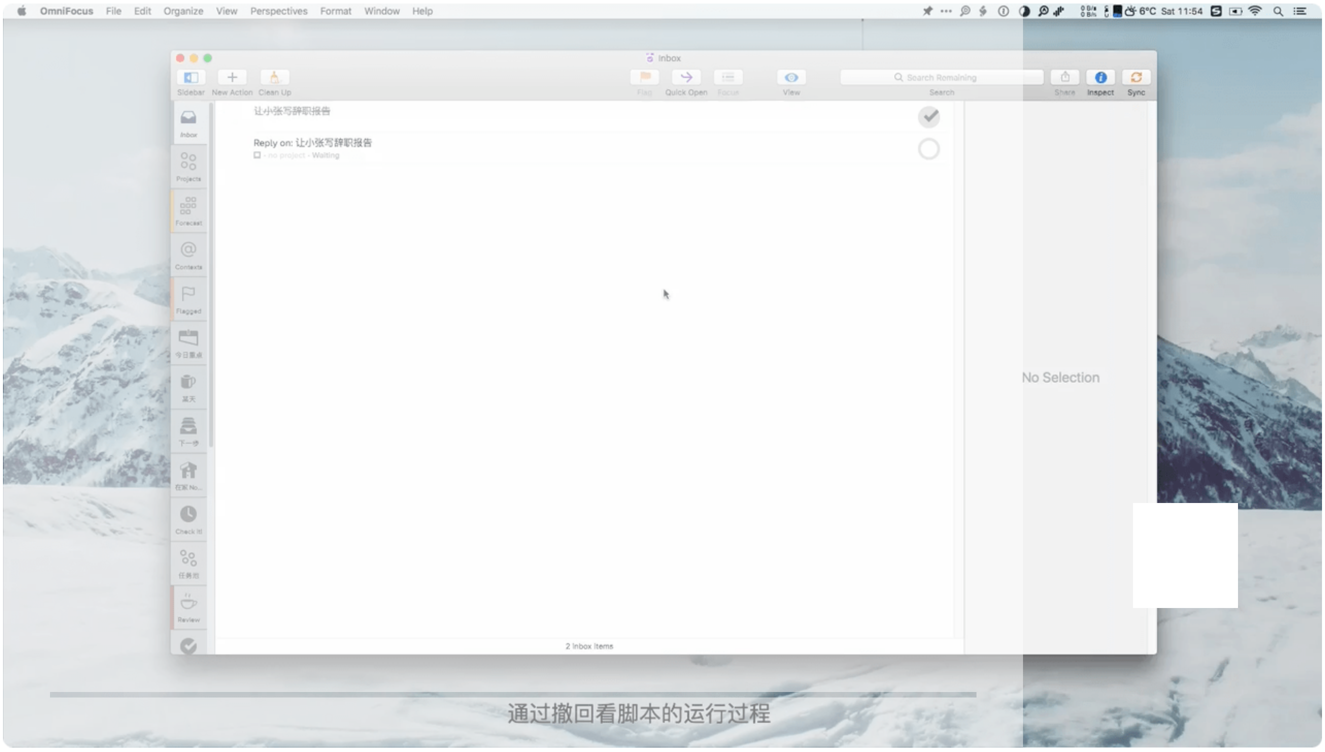
以「Complete and Await Reply（完成并等待回应）」（点击下载）这个脚本为例，来看看 AppleScript 在 OmniFocus 中到底是怎么运行的。

生活中的有些任务，虽然本身已经完成，但需要我们隔一段时间后再次检查它的成果。为了新家你可能在淘宝上买了一批家具或者饰品，当你下单的时候，其实「买床头灯」、「买墙纸」等任务已经完成，所以最好是及时勾掉任务。但有时会因为一次性买的东西太多，忘了到底有哪些东西到货了，有哪些还没到，如果每次都要进淘宝查看订单，就显得繁琐。

面对这种需要再次检查的任务，就需要「Complete and Await Reply」脚本。**选中**一个任务后，**运行**脚本，被选中的动作就会被设为**已完成**。同时在项目下方**新建**一个同名动作，只是会加上「**Reply on:**」的前缀，就像回复邮件时一样。这样当床头灯到货了，再去勾掉这个带有 Reply on 的任务即可，剩余有哪些没到货的，就一目了然。


Open Scripts Folder
Open Script Editor
OmniFocus Scripts
Append Note to Selected Task
Clear Dates
Complete and Await Reply
defer tasks
Focus in New Tab
Later
Open Links
Populate Template Placeholders
Prepare Task Completion Report 2
Total Time
DEVONthink Pro
Get files name
OmniFocus
Paste URL from Safari Tabs
让小张写辞职报告
1 Inbox Item

而当你按下「撤销」（⌘Command - Z）时，就能看到这个脚本在背后运行时，OmniFocus 到底发生了什么。完成动作和新建动作并不是一口气发生的，脚本第一步将老的动作**设为已完成**，再**复制**了一个一模一样的动作，最后**修改**动作的标题，**加上**「Reply on:」。


让小张写辞职报告
Reply on: 让小张写辞职报告
No Selection
2 Inbox Items

通过撤回看脚本的运行过程

通过这个脚本可以看出，AppleScript **并没有为软件新增功能。**只是通过代码，实现了我们原先需要自己手动完成的动作。就像生产线上的机械手，把每一步的流程都固定下来，可以重复运行。同时也能看到，AppleScript 是可以**直接操作 macOS 应用**的，也就是定义中所谓的「control scriptable Macintosh applications」。而其他脚本语言，例如 zsh，甚至强大如 Python，都不能直接控制 Mac 中的图形应用。

不过既然脚本只是设定了一套固定的流程，功能上并没有增加，那为什么我们要使用晦涩的 AppleScript，而不停留在图形界面下使用 OmniFocus 呢？

为什么我们要使用 AppleScript 来操作 OmniFocus ？

世上本无图形界面程序，不会使用命令行的人多了，就自然有了图形界面程序。但是想要极致高效的操作程序，甚至为已有的程序增加一些新的功能，就还是要靠 AppleScript。

自动化操作提高操作效率

像所有的脚本语言一样，使用 AppleScript 的第一个好处在于：面对大量机械重复的动作时，脚本的自动化能**节省我们操作的时间。**

OmniFocus 是以「项目」为单位来组织任务的，而实际上一份工作所需要处理的项目，本质上流程是相对固定的。例如我在少数派写文章的流程，大抵是如下的：

- 搜集资料，组织导图
- 完成大纲
- 完成初稿
- 修改文章
    - 检查段落逻辑
    - 修改语言风格
    - 完成配图
    - 上传网站后台

每次为了新的文章创建项目时，所要创建的动作都是基本类似的。那为什么我每次都要输入同样的内容，而不能自动生成一个新的文章项目，最多修改下文章名和截止日就可以了呢？由 Curt Clifton 编写的 「Populate Template Placeholders」脚本（点击下载），就能实现**根据模板创建新的项目**，只需要手动修改项目的关键信息和推迟至、截止日期即可。

Populate Template Placeholders（根据模板创建项目）

要想成功运行这个脚本，需要我们先在 OmniFocus 中做一些准备工作：

- 在 OmniFocus 像正常创建项目一样，**创建一个项目模板。**这个模板就是脚本用来创建项目的基础，所以就按照实际需求创建即可。Curt Clifton 是特地新建了一个「Template」文件夹，用来存放这些项目模板。但实际上经过我测试，项

目所在的文件夹并不影响脚本运行，完全可以根据你自己的喜好存放。

- 将你需要修改的信息**设定为占位符**。项目中有些信息是需要根据项目不同来调整的，例如文章模板中的文章名。所以我的模板项目的项目名是 `>文章名<` ，其中 `>` 和 `<` 用来标记需要修改的内容。需要注意的是，原始脚本中使用的是 `«»` 来标记，我为了输入方便，将脚本的下面两行：

  ```
  property placeholderOpener : "«"
  property placeholderCloser : "»"
  ```

  修改为：

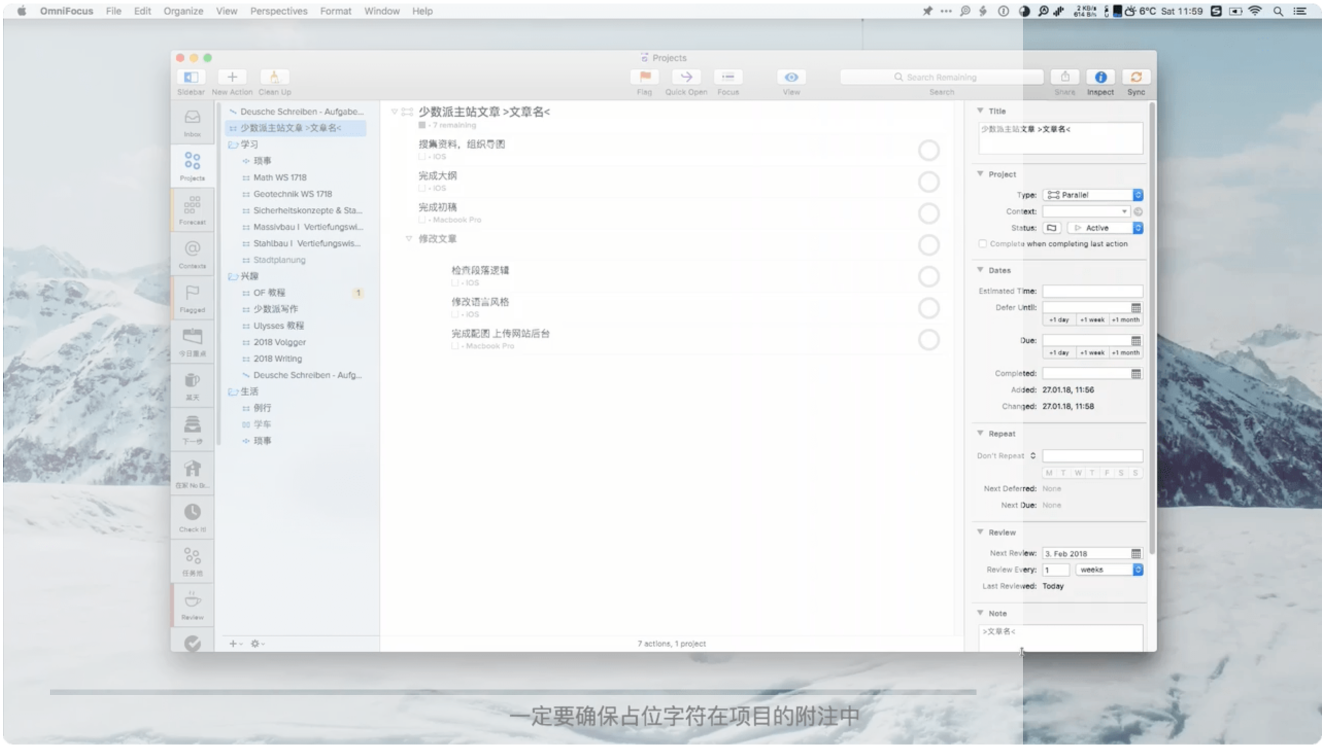
  ```
  property placeholderOpener : ">"
  property placeholderCloser : "<"
  ```

- 设定好项目所有动作中出现的占位符后，将它们全部**罗列到项目的附注里**（注意！不是某个动作的附注）。值得留意的是，占位符一定位于附注的最后一行。如果你在主大纲视图里看到下面这种情况，很可能附注最后一行被回车占据，这会导致脚本运行失败。

- **为项目设定推迟至日期和截止日期**。这一步不是必 所以根据自己的需求来设定。值得注意的是，如果给项目只设定了推迟至日期，脚本运行时可以输入新的推迟至日期；如果只设定了截止日期或者同时设置了两者，脚本运行时**只能**修改截止日期。

准备工作完成后，我们就可以运行这个脚本看看实际效果。

1. **选中项目模板**（在工具条或主大纲中选中，不要只选中其中的动作），**运行脚本**。
2. 根据对话框依次**输入要修改的信息**。脚本会先编辑让我们占位符的内容，例如文章名（取决去你给项目设定了多少占位符），最后修改推迟至日期或截止日期。
3. 脚本自动在项目列表最下方创建出新项目，别忘了把它拖到相应的文件夹。


OmniFocus
File
Edit
Organize
View
Perspectives
Format
Window
Help
Projects
Sidebar
New Action
Clean Up
Flag
Quick Open
Focus
View
Search Remaining
Search
Share
Inspect
Sync
Inbox
Projects
Forecast
Contexts
Flagged
今日重点
某天
下一步
Check In
Review
Deusche Schreiben - Aufgabe...
少数派主站文章 >文章名<
学习
琐事
Math WS 1718
Geotechnik WS 1718
Sicherheitskonzepte & Sta...
Massivbau I Vertiefungswi...
Stahlbau I Vertiefungswis...
Stadtplanung
兴趣
OF 教程
少数派写作
Ulysses 教程
2018 Volgger
2018 Writing
Deusche Schreiben - Aufg...
生活
例行
学车
琐事
7 actions, 1 project

少数派主站文章 >文章名<
· 7 remaining
搜集资料，组织导图 · IOS
完成大纲 · IOS
完成初稿 · Macbook Pro
修改文章
检查段落逻辑 · IOS
修改语言风格 · IOS
完成配图 上传网站后台 · Macbook Pro

Title: 少数派主站文章 >文章名<
Project
Type: Parallel
Context:
Status: Active
☐ Complete when completing last action
Dates
Estimated Time:
Defer Until: +1 day +1 week +1 month
Due: +1 day +1 week +1 month
Completed:
Added: 27.01.18, 11:56
Changed: 27.01.18, 11:58
Repeat
Don't Repeat
Next Deferred: None
Next Due: None
Review
Next Review: 3. Feb 2018
Review Every: 1 weeks
Last Reviewed: Today
Note: >文章名<

一定要确保占位字符在项目的附注中

使用「Populate Template Placeholders」，我就不再需要每次手动输入「收集资料」这种动作，所有把这些重复性的工作让脚本帮我完成。使用完你会有一种感觉：

难道这些事本来就不应该通过机器完成吗？

AppleScript 的魔力正在于此。

拓展 OmniFocus 的功能

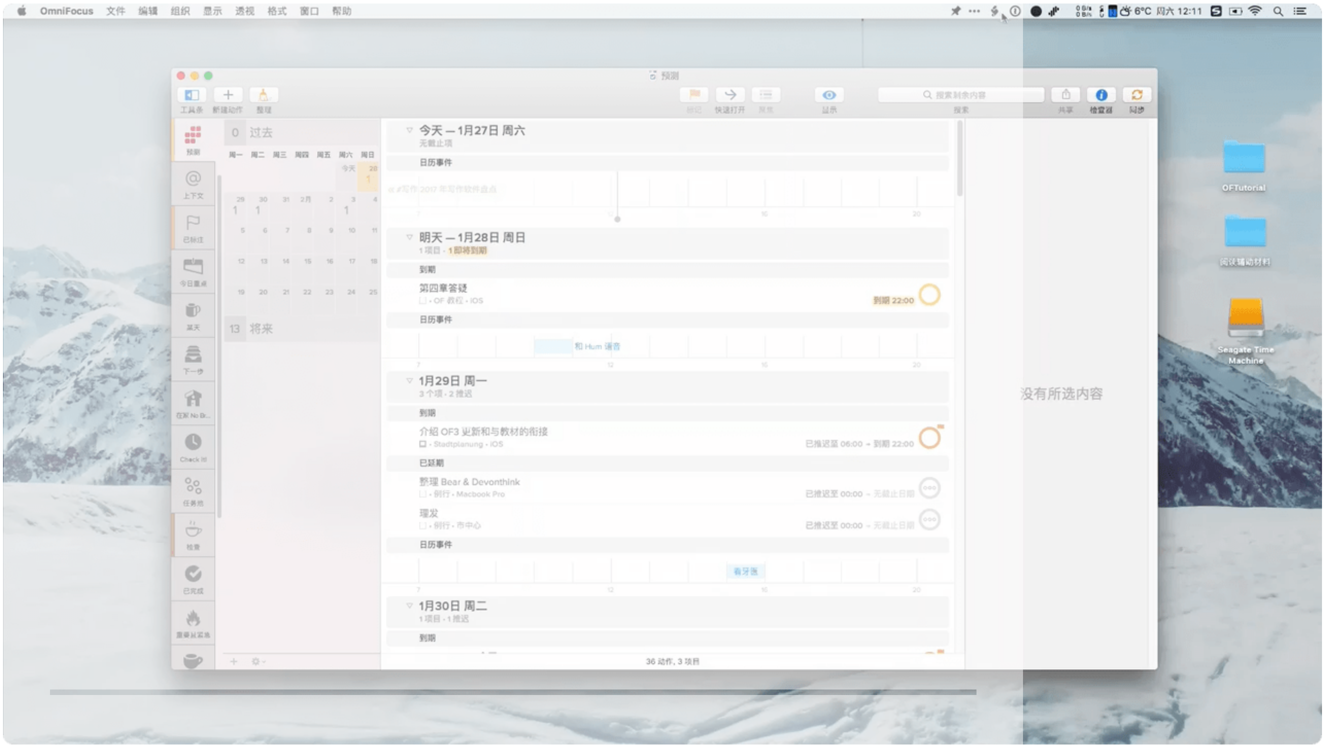
之前我一直强调，AppleScript 是将应用现有的功能组合起来，不是去开发新的功能。例如「Populate Template Placeholders」脚本所做的事，其实手动一步步也能完成。但别忘了 AppleScript 定义中的「as well as parts of macOS itself」，这说明 AppleScript 也能调动一些系统本身的功能，虽然不能将 OmniFocus 变成浏览器或者播放器，但也**能在系统范畴内拓展 OmniFocus 的功能。**

任务管理一贯强调回顾和检查，在第三章《将任务从 OmniFocus 中拿出来》中，曾介绍过「已完成」透视，可以查看已完成的动作和项目。但这个视图中会显示数据库中过往所有的动作，在我写自己的周总结时，其实只想看到上一周我完成了哪些动作。

Weekly project report generator（周报生成器）

Finlay Boo 编写的 「Weekly project report generator」脚本（点击下载），就能自动生成一个 Markdown 文件，里面记录了活跃的项目和上一周完成的动作列表。

使用这个脚本非常简单，甚至不需要先打开 OmniFocus，可以利用脚本菜单来直接运行。运行后会弹出对话框提示你 一份 markdown 文本，也就是我们想要的周报。每周写周总结的时候，运行一次这个脚本，再将 md 文本拖到 OmniOutliner 中作为参考资料，免得费劲地在 OmniFocus 中查找上周完成的动作。


今天 — 1月27日 周六
明天 — 1月28日 周日
1月29日 周一
1月30日 周二
没有所选内容
OFTutorial
Seagate Time Machine

「Weekly project report generator」脚本就是一个绝佳的范例，展示了如何利用系统内置的一些功能接口，例如拷贝文本、创建文档，拓展了 OmniFocus 的使用场景。

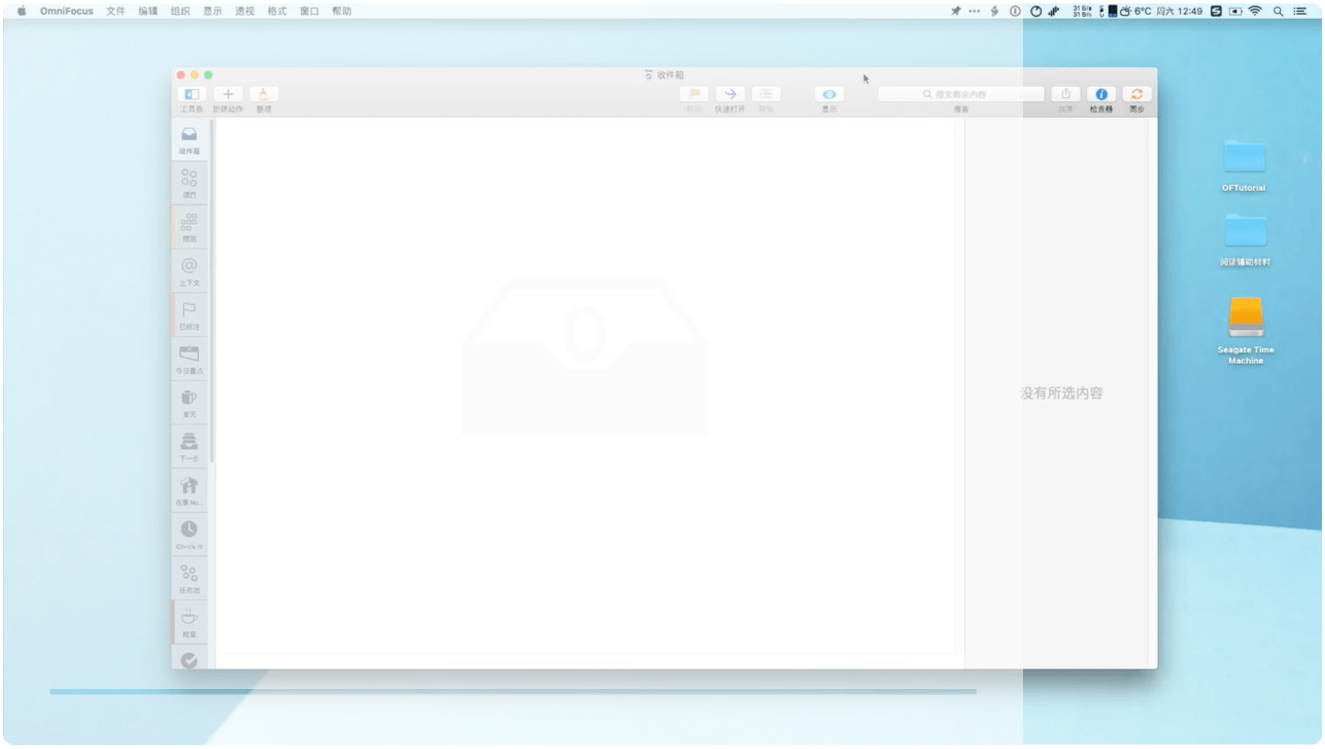
感性认识 AppleScript 在 OmniFocus 中的实现

从第一章开始，我们其实就不断在接触不同的 OmniFocus 脚本，这一章又通过三个较为复杂的脚本，了解了 AppleScript 的本质和优势，相信对于大部分读者来说，已经初步了解 AppleScirpt 到底在 OmniFocus 是如何发挥作用的，使用 AppleScript 也不成问题。

接下来我们将进一步深入，依然是以优秀的 AppleScrip 列，去看看脚本内部的精妙之处。限于篇幅，我们不会一行行代码去分析脚本，只点出脚本实现的关键之处，以期达到抛砖引玉的效果。

Send Safari URL to OmniFocus（发送 Safari URL 到 OmniFocus）

在收集任务阶段，常常有的动作就来自某个网页。OmniFocus 本身也提供了相应的功能，在 Safari 的共享菜单中。但使用这个功能时，需要自己手动添加动作名，非常不人性化。由 Daniel Compton 编写的 「Send Safari URL to OmniFocus」脚本（点击下载）就能解决这个痛点。


OmniFocus
文件
编辑
组织
显示
透视
格式
窗口
帮助
收件箱
没有所选内容
OFTutorial
阅读辅助材料
Seagate Time Machine

和「Weekly project report generator」一样，运行这个脚本时不需要打开 OmniFocus，直接在脚本菜单中运行即可。运行成功后 OmniFocus 中的收件箱就会出现一个新的动作，标题是网页内的文章名（这比共享菜单创建的动作不知道高到哪里去了），而附注里就是网址链接。

（在脚本编辑器中）打开脚本，会发现作者仅用 7 行就完成了这个功能：

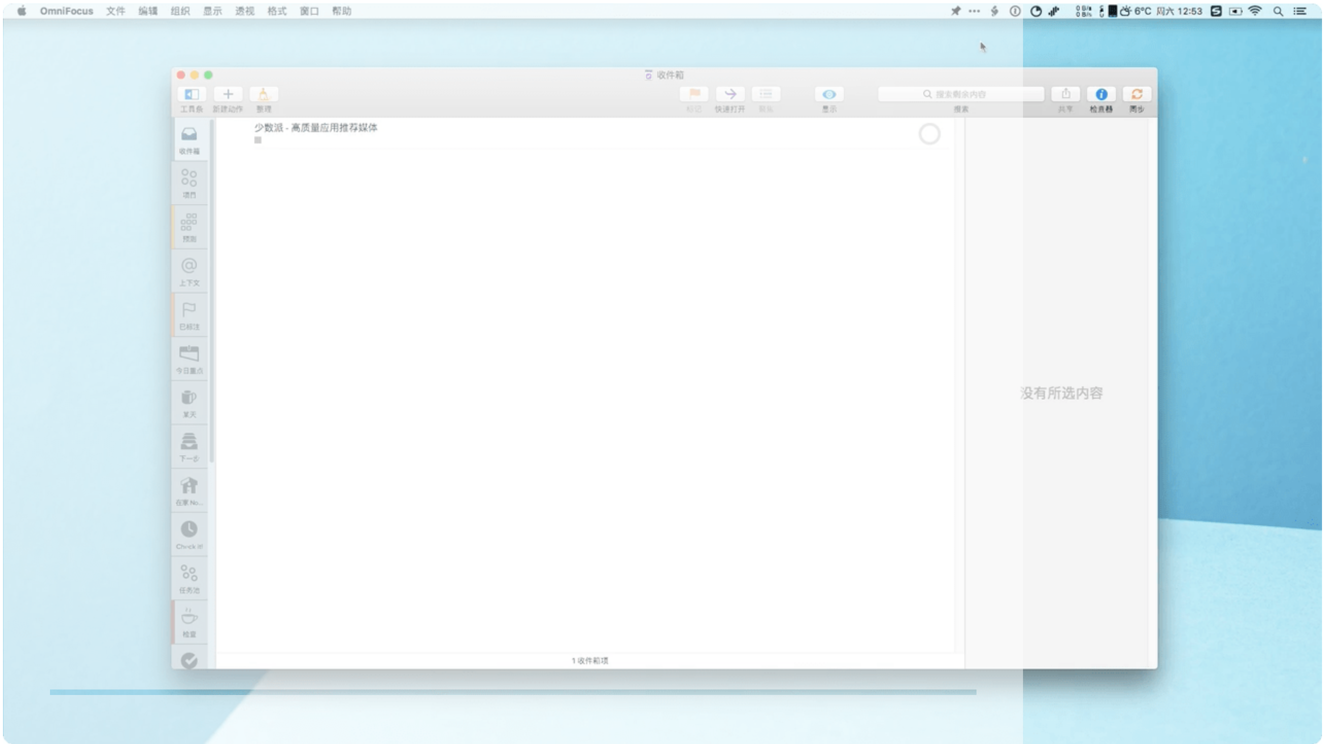
```
tell application "Safari"
    set theURL to (URL of current tab of window 1)
    set theName to (name of current tab of window 1)

end tell


tell front document of application "OmniFocus"
    make new inbox task with properties {name:(theName), note:theURL}
end tell
```

第一部分（前四行）通过 Safari 获取了当前网页的 URL 和标签名，第二部分则是在 OmniFocus 中创建了一个新的收件箱动作，并且将 URL 设为动作属性中的附注，标题名设置为标签名。思路非常直接，但充分发挥了 Safari 和 OmniFocus 内置的关键词。

Open Links（直接打开动作中的链接）

2Do 中对任务可以进行「操作」。其中有个很实用的动作是：如果给任务附上了网站链接，可以直接点击打开网站，而 OmniFocus 则要自己手动（通常在附注中）点击网址，显得有些原始。联系到上面介绍的保存 Safari URL 到 OmniFc 的脚本，为什么我们不用一个脚本，来实现点击动作直接打[illegible]网页的效果呢？

「Open links」（点击下载）就可以实现**直接打开动作内网页链接**的功能。运行脚本前需要先选中动作，再运行。但我们之前通过「Send Safari URL to OmniFocus」脚本保存的动作是将网址下在附注中，而「Open links」脚本则要求链接是保持在动作标题中，而且不能有其他非网址的内容。这就需要我们自己对脚本进行一下改造了。



打开「Open links」脚本可以看到这个实现的关键部分在于：

```
tell application "OmniFocus"
    set _url to _action's name
    do shell script "open " & _url's quoted form
end tell
```

原始脚本将动作的标题名（ `name` ）赋值给了变量 `_url` ，下一行通过 `do shell` 语句，在 Safari 中打开前面赋值语句提取到的内容（即网址）。所以如果想让这两个脚本配合使用，要做的就

非常简单，只需要将动作的附注，而不是标题名，赋值给变量 `_url` 就可以了。

```
tell application "OmniFocus"
    set _url to _action's note
    do shell script "open " & _url's quoted form
end tell
```

其中 `note` 是 OmniFocus 自带的属性，表示附注。这样「Open Links」脚本就会打开动作附注中的内容。

Send Safari Tab List to OmniFocus（保存 Safari 所有的标签页到 OmniFocus）

通常在写论文查资料时，我会一口气在 Safari 中打开好几个标签页。但写论文不可能一天进行完，经常头一天写到一半，隔天再继续写。但 Safari 的网页通常不会特意保留下来，一般是随手就关掉了。

「Send Safari Tab List to OmniFocus」（点击下载）这个脚本，就可以一口气将目前 Safari 打开的所有网页，全部保存在一个 OmniFocus 收件箱动作中。

少数派
专题广场
Matrix
付费栏目
效率工具
手机摄影
生活方式
游戏
硬件
年度征文
「2017 年，这些工具让我摆脱拖延症」
点击参与
下载科普
隐私、稳定与速度
LaunchBar
全方位优化你的 macOS 使用习惯
Workflow Gallery 回来了！
MacDropAny：简单三步，把你的任何文件夹变成「云盘」！
App+1
Matrix精选
我的数字 2017：Notion、笔记和其他
Homebrew：命令行效率工具的敲门砖
[iOS开发经历]增强现实入坑全流程
"全面屏"
我的VPS使用笔记之：新机基础配置

和「Send Safari URL to OmniFocus」很类似，「Send Safari Tab List to OmniFocus」脚本也可以直接运行。运行后在 OmniFocus 收件箱中生成一个动作，附注里保存了所有 Safari 标签名和网址。这样下次重新进入工作状态时，可以先打开之前保存好的网页。

查看脚本会发现，这个脚本的运行逻辑基本是「Send Safari URL to OmniFocus」一样，先通过 Safari 获取网址和标签名，再写入 OmniFocus 的动作中。

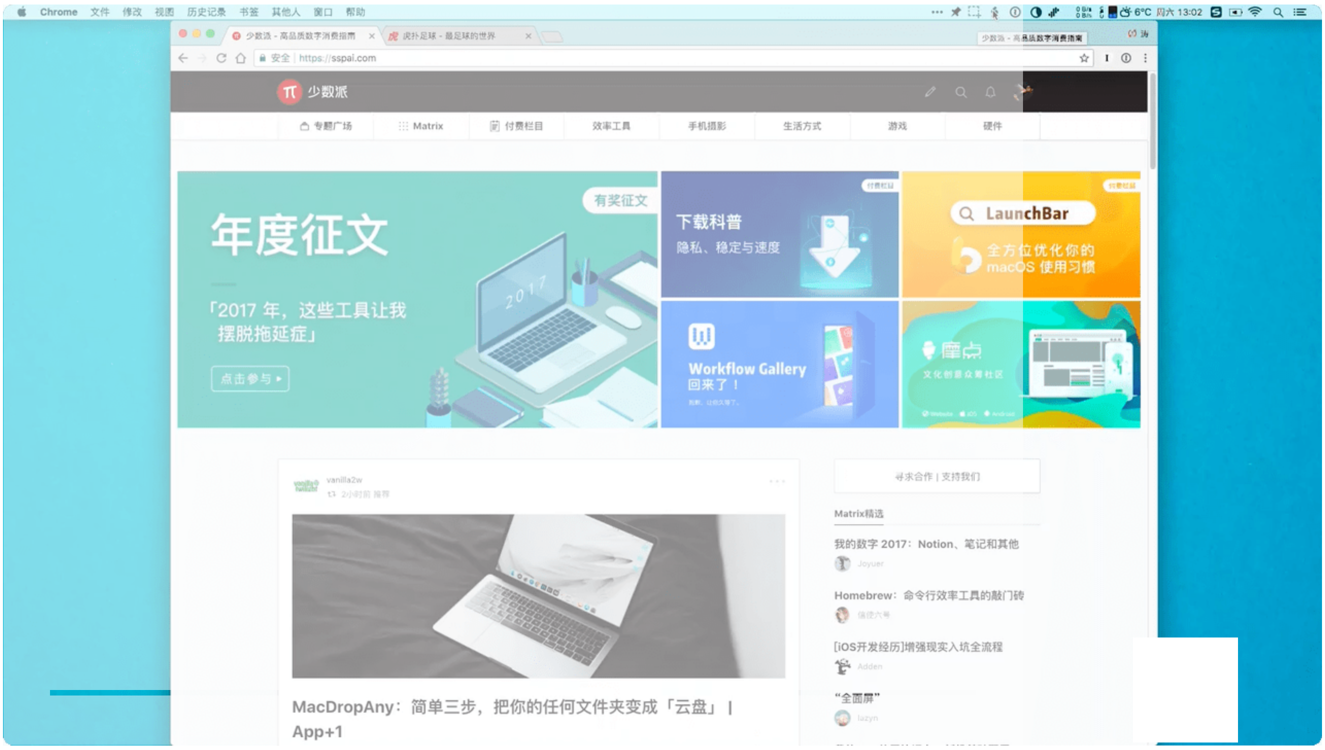
```
repeat with t in (tabs of safariWindow)
    set currentTab to currentTab + 1
    set tabTitle to (name of t)
    set tabURL to (URL of t)
    ...
end repeat
```

而和之前脚本不同的是，为了获取所有的 Safari 标签，作者使用了 `repeat` 循环语句，依次获取 Safari 中的每一个 tab 的信息。在将信息写入到 OmniFocus 时，其实使用了类似的语句：

```
make new inbox task with properties {name:(noteTitle), note:urlList as text}
```

除了主体脚本之外，作者还给这个脚本加上了通知功能，根据所获取的标签数的不同，利用 `if` 控制语句显示不同的信息。其中关键的命令是使用了 AppleScript 命令 `display notification`，用于在通知中心中显示通知。

这个脚本还有类似的 Chrome 版本（点击下载）。


Chrome
文件
修改
视图
历史记录
书签
其他人
窗口
帮助
少数派 - 高品质数字消费指南
虎扑足球 - 最足球的世界
https://sspai.com
少数派
专题广场
Matrix
付费栏目
效率工具
手机摄影
生活方式
游戏
硬件
有奖征文
年度征文
「2017 年，这些工具让我摆脱拖延症」
点击参与
下载科普
隐私、稳定与速度
LaunchBar
全方位优化你的 macOS 使用习惯
Workflow Gallery
回来了！
寻求合作 | 支持我们
Matrix精选
我的数字 2017：Notion、笔记和其他
Homebrew：命令行效率工具的敲门砖
[iOS开发经历]增强现实入坑全流程
“全面屏”
MacDropAny：简单三步，把你的任何文件夹变成「云盘」 | App+1

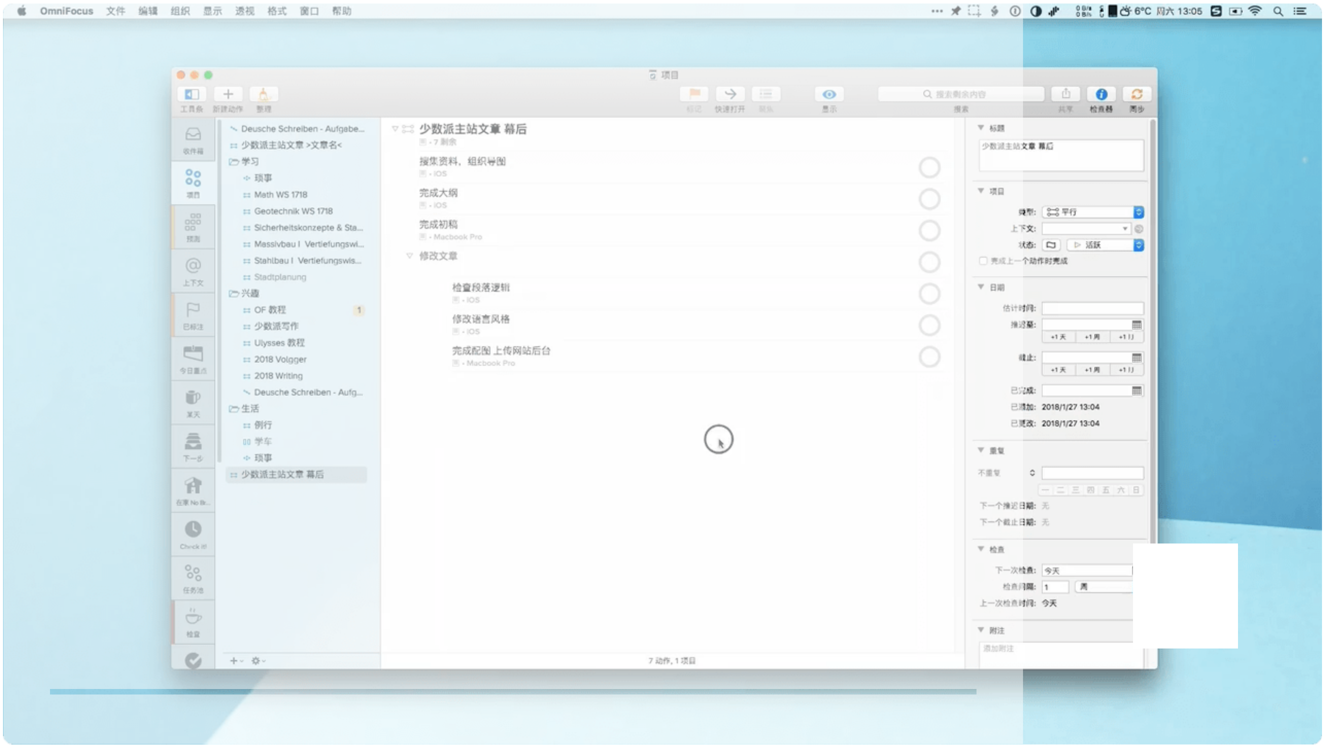
相比 Safari 的版本，其实 Chrome 版本的脚本改动非常小。打开脚本代码后会发现，除了变量名称的差别，真正的区别在于将 `tell application "Safari"` 变成了

`tell application "Google Chrome"` ，其余的运行逻辑则完全一致。

Get total time of selected items（计算动作所需的总共时间）

在动作属性里可以设置动作的估计时间，如果要预估完成若干动作所需的所有时间，只需要统计它们的估计时间即可。但是 OmniFocus 并不会显示若干动作合起来总共的估计时间。这时候又到了 AppleScript 出马的时候了。

由 Dan Byler 编写的 「Get total time of selected items」（点击下载），就能轻松地实现统计动作的估计时间的功能。选中想统计的动作后，运行脚本即可，脚本会弹出对话框告诉你总计的估计时间。


OmniFocus
少数派主站文章 幕后
搜集资料、组织导图
完成大纲
完成初稿
修改文章
检查段落逻辑
修改语言风格
完成配图 上传网站后台

这个脚本实现起来其实思路比较清晰，依次获取每个动作的估计时间，再相加即可。而在脚本中 Dan Byler 是这样实现的：

```
repeat with thisItem in validSelectedItemsList
    set thisEstimate to estimated minutes of thisItem
    if thisEstimate is not missing value then set totalMinutes to totalMinutes + thisEstimate
end repeat
```

依然是用 `repeat` 循环获取动作信息。每获取一个新的动作估计时间（ `estimated minutes` ），就将它赋值给变量 `thisEstimate` ，再与之前计算过的 `totalMinutes` 相加，得到新的 `totalMinutes` (此时 totalMinutes 的值已经改变了）。

此外作者还将脚本实现的非常完善。例如将总共的统计时间换算成小时和分钟：

```
set modMinutes to (totalMinutes mod 60)
set totalHours to (totalMinutes div 60)
```

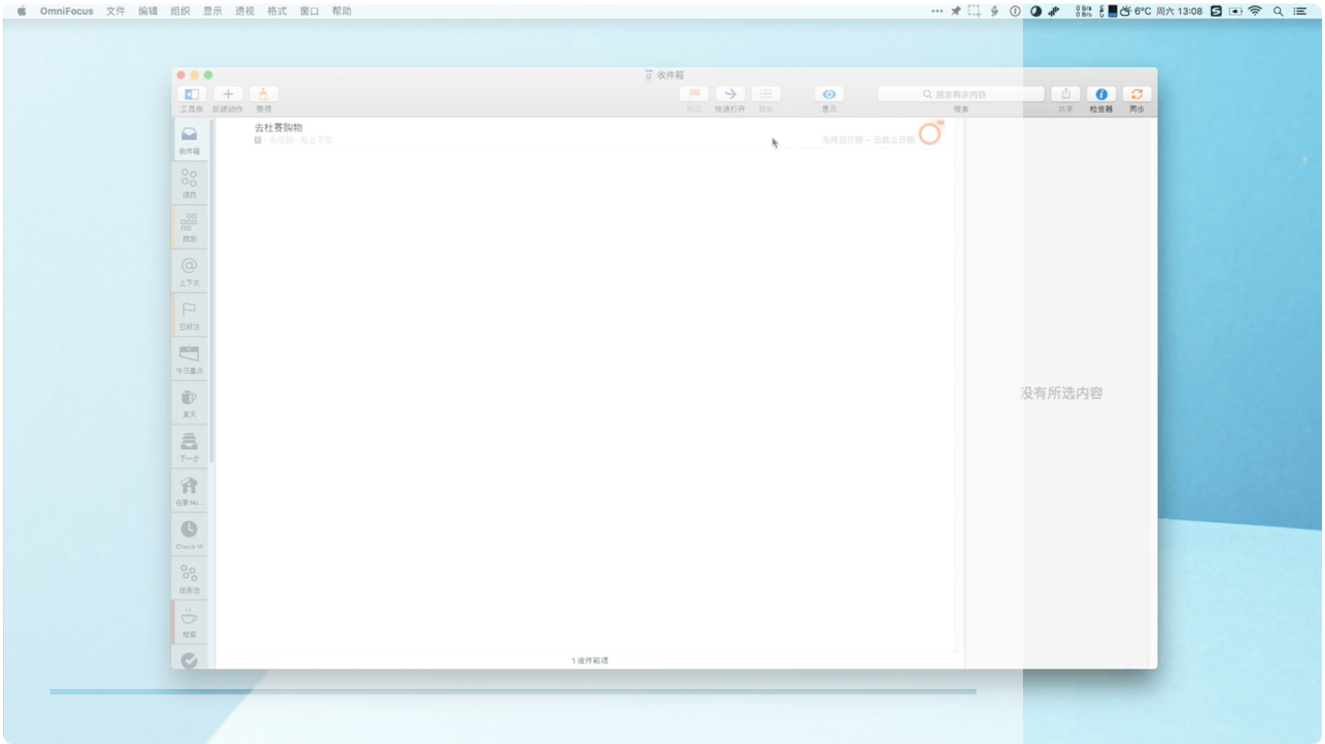
作者还使用了 Growl 来显示通知信息，相比系统自带的通知组件能显示更多的信息。脚本中其实大部分的代码都是在设置通知信息，其实核心功能的代码在 `main()` 中就实现了。

Defer OmniFocus tasks to random dates（在 Mac 上给动作随机设置推迟至日期）

在 iOS 版的 OmniFocus 中，有一项功能 Mac 版居然没有的——随机设置推迟至日期。当我们并不明确的知道某个项目或动

作，应该进入可执行状态的时间时，这项功能就可以快速地随机设定一个时间。然而在 Mac 版上并没有这个功能，之前我们介绍过的「Later」脚本，也是要指定一个时间间隔。

「Defer OmniFocus tasks to random dates」脚本（点击下载）就给 Mac 版的 OmniFocus 也带来了同样的功能，使用甚至比 iOS 版还要直接。选中动作后运行脚本，脚本就会自动给动作设置一个推迟至时间。这个时间的范围会在 14 到 90 天之间，作者之所以这么设置，是因为 iOS 版的随机时间范围也是在 14 到 90 天之间。


OmniFocus
文件
编辑
组织
显示
透视
格式
窗口
帮助
6°C 周六 13:08
收件箱
工具条
新建动作
整理
标记
快速打开
聚焦
显示
搜索剩余内容
搜索
共享
检查器
同步
去杜赛购物
无项目 · 无上下文
无推迟日期 – 无截止日期
收件箱
项目
预测
上下文
已标注
今日重点
某天
下一步
在家 No...
Check in
任务池
检查
没有所选内容
1 收件箱项

打开脚本可以看到，这个脚本的实现由几个 Handler 组合完成。Handler 可以理解为其他编程语言的函数。 `myDeferDate()` 是这个脚本中的关键部分，利用 `random number` 命令生　个随机数来表示天数：

```
set random_days to random number from lower_limit to
 upper_limit
```

最后经过转换，它会返回一个随机的日期。随后在 `calculateDate()` 函数中，它要求接收所选动作的推迟日期，并返回 `myDeferDate()` 生成的日期。在 `processAction()` 函数中，将所选动作的推迟至日期设定为 `calculateDate()` 的返回值。 `selectedActions()` 会返回一个包含所有动作信息的列表，最后利用 `repeat` 循环，对列表内每一个动作进行运行。

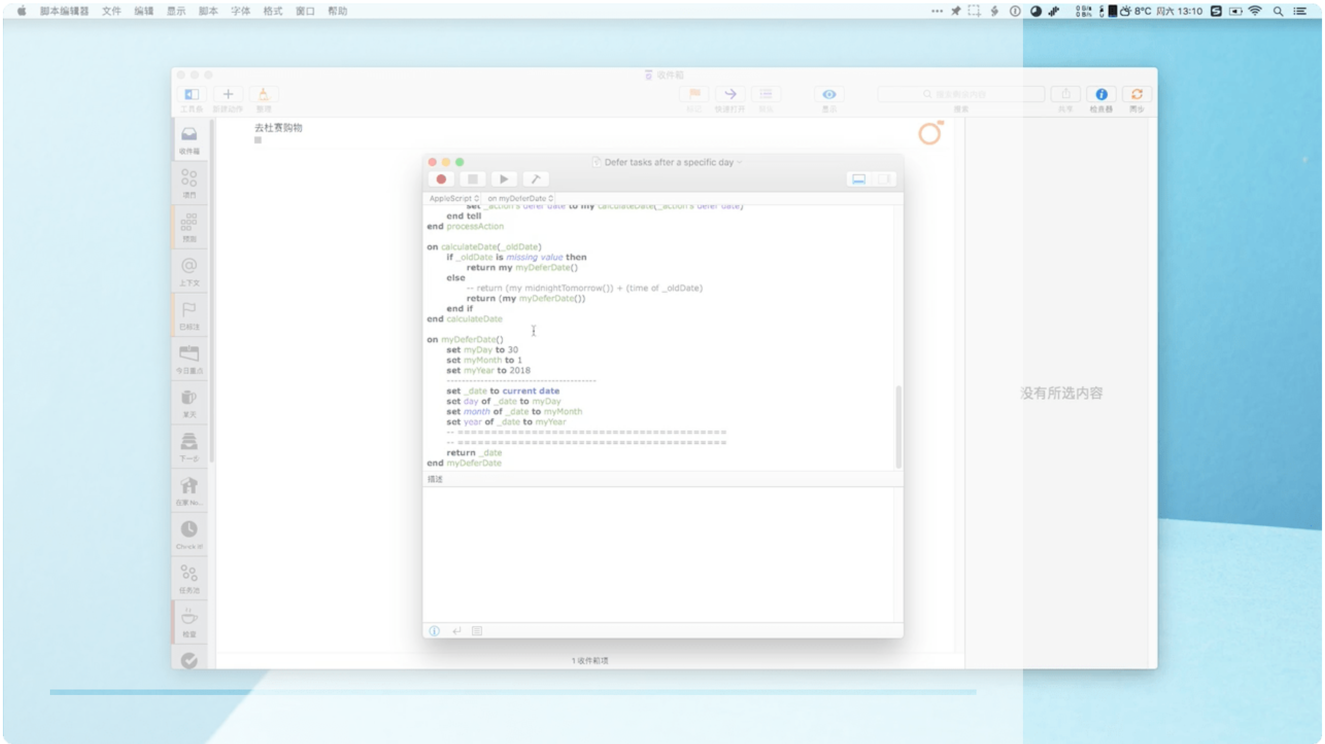
利用 Handler 编写脚本的好处在于，可以实现软件设计所常追求的「高内聚低耦合」。通俗来将，就是代码的功能模块进行了划分，尽可能互相不影响。这样修改其中一个模块的功能，就可以改变代码的作用。

例如这个脚本中的 `myDeferDate()` 设定了推迟日期的算法。如果我们不像随机的设定日期，而是一个特定的日期，我们只需要修改 `myDeferDate()` 函数的算法，就可以改变脚本的功能。例如下面这种实现：

```
on myDeferDate()
    set myDay to 30
    set myMonth to 1
    set myYear to 2018
    set _date to current date
    set day of _date to myDay
```

```
        set month of _date to myMonth
        set year of _date to myYear
        return _date
    end myDeferDate
```

`myDeferDate()` 最后是要返回变量 `_date` ，于是我们直接设定变量的值为 2018 年 1 月 30 号，这样返回值就会是对应的日期，这个脚本的功能也就从随机推迟变成了推迟到特定日期。而脚本中的其他函数都不需要改变。



```
        set _action's defer date to my calculateDate(_action's defer date)
    end tell
end processAction

on calculateDate(_oldDate)
    if _oldDate is missing value then
        return my myDeferDate()
    else
        -- return (my midnightTomorrow()) + (time of _oldDate)
        return (my myDeferDate())
    end if
end calculateDate

on myDeferDate()
    set myDay to 30
    set myMonth to 1
    set myYear to 2018
    ------------------------------------
    set _date to current date
    set day of _date to myDay
    set month of _date to myMonth
    set year of _date to myYear
    -- ========================================
    -- ========================================
    return _date
end myDeferDate
```

AppleScript 的组成

AppleScript 作为一种编程语言，所有的组成也无外乎程[illegible]言的四种成分：**数据成分、运算成分、控制成分、传输成分**。上文在展示这些实用的脚本的同时，其实也依次展示了 AppleScript 的这四种成分。

数据成分指的是语言中数据的类型，例如数值、列表等。通过赋值语句 `set A to B` ，可以将各种数据赋予给变量。例如 `set _url to _action's name` 这句，就将一串链接赋给了变量 `name` 。

运算成分指的是语言中包含的四则或则逻辑运算，通过运算符来修改变量的值。 `set totalMinutes to totalMinutes + thisEstimate` 就是利用运算符 `+` 来进行累加计算。

控制成分则是程序语言的关键，所有的程序语言都会必须有条件判断、循环执行语句[2] ，而 AppleScript 也有相应的 `if` 和 `repeat` 语句（当然不止这两句，这里受制于篇幅不再展开）。

传输成分则用来表达程序中数据的传输。在「Defer OmniFocus tasks to random dates」这个脚本中我们可以看到，脚本中的函数传入一个数值，再传出一个数值，可以修改了变量的值，也可以创建新的返回值。

AppleScript 作为编程语言有其复杂性，但是所有的代码都不外乎以上提到的四种成分。这里特地提出来，希望各位读者阅读时不必过于恐慌。

向更深层次进发

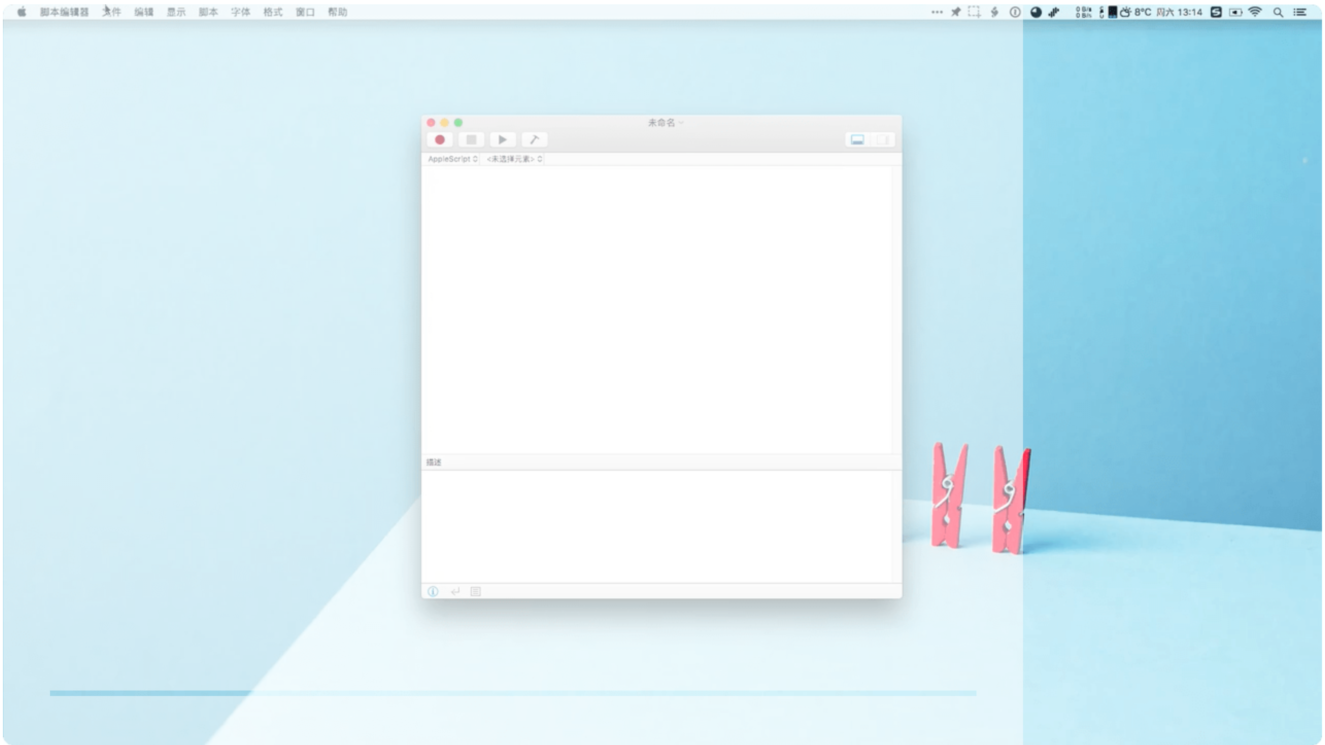
受制于内容和篇幅，本章没有继续深入讲解 AppleScript 的编写，更多精力放在了使用、阅读和修改 AppleScript 上。为了让

有兴趣的读者继续自行深入研究 AppleScript，这里提供一些辅助材料。

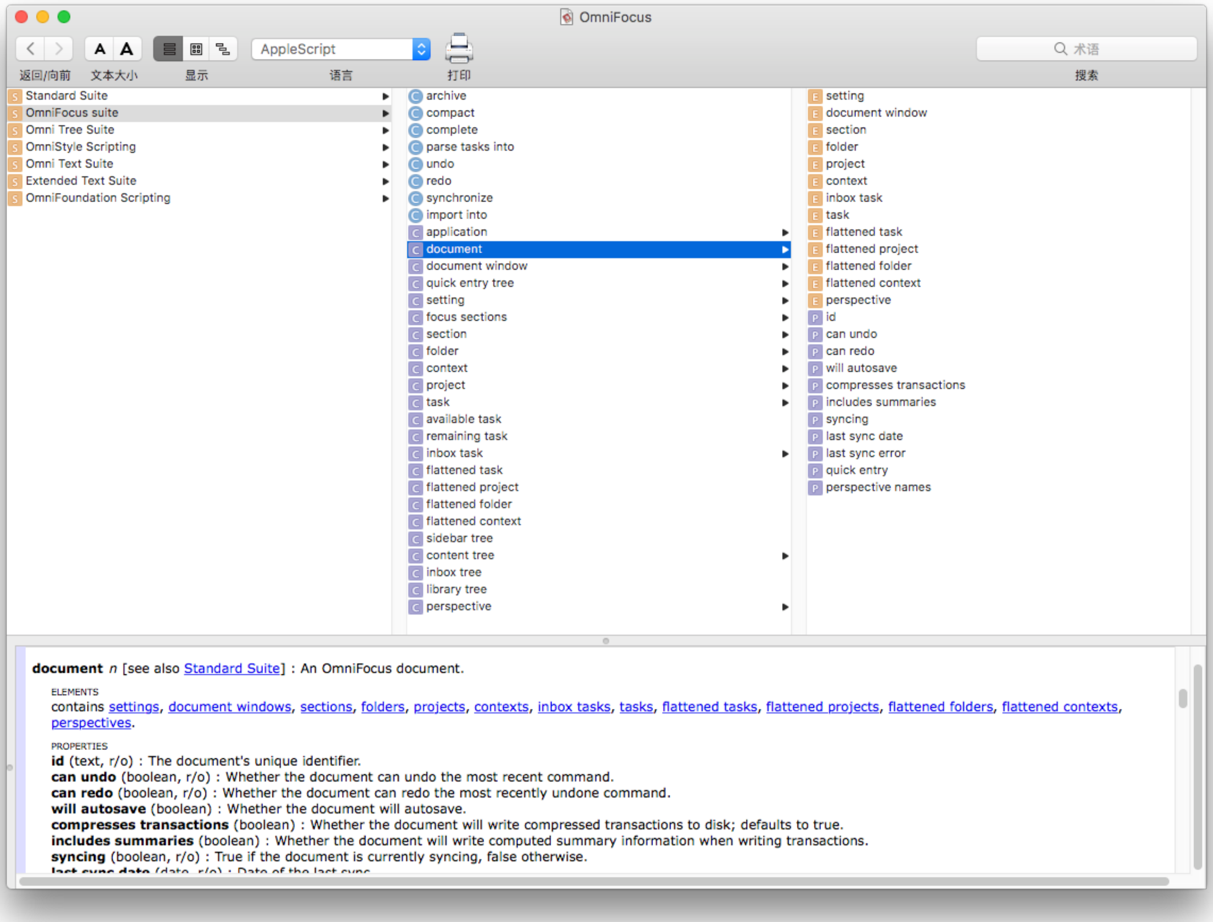
AppleScript editor 字典

对于想有心自行编写 AppleScript 的读者来说，字典是必须首先会用的工具。

打开脚本编辑器后，点击「文件 - 打开词典」，就会弹出选框，选中 OmniFocus 即可显示 OmniFocus 自带的所有的套装、类、命令和属性。


脚本编辑器
文件
编辑
显示
脚本
字体
格式
窗口
帮助
未命名
AppleScript
<未选择元素>
描述

套装是开发者将一系列相关的类、命令和属性打包在一起。例如 OmniFocus 有专属的 `OmniFocus Suite` ，同时也有 Or 系应用通用的 `Omni Tree Suite` 。套装内的内容多少，完全有开发者决定开放多少。


OmniFocus
返回/向前
文本大小
显示
AppleScript
语言
打印
术语
搜索
Standard Suite
OmniFocus suite
Omni Tree Suite
OmniStyle Scripting
Omni Text Suite
Extended Text Suite
OmniFoundation Scripting
archive
compact
complete
parse tasks into
undo
redo
synchronize
import into
application
document
document window
quick entry tree
setting
focus sections
section
folder
context
project
task
available task
remaining task
inbox task
flattened task
flattened project
flattened folder
flattened context
sidebar tree
content tree
inbox tree
library tree
perspective
setting
document window
section
folder
project
context
inbox task
task
flattened task
flattened project
flattened folder
flattened context
perspective
id
can undo
can redo
will autosave
compresses transactions
includes summaries
syncing
last sync date
last sync error
quick entry
perspective names

document *n* [see also Standard Suite] : An OmniFocus document.

ELEMENTS
contains settings, document windows, sections, folders, projects, contexts, inbox tasks, tasks, flattened tasks, flattened projects, flattened folders, flattened contexts, perspectives.

PROPERTIES
id (text, r/o) : The document's unique identifier.
can undo (boolean, r/o) : Whether the document can undo the most recent command.
can redo (boolean, r/o) : Whether the document can redo the most recently undone command.
will autosave (boolean) : Whether the document will autosave.
compresses transactions (boolean) : Whether the document will write compressed transactions to disk; defaults to true.
includes summaries (boolean) : Whether the document will write computed summary information when writing transactions.
syncing (boolean, r/o) : True if the document is currently syncing, false otherwise.
last sync date (date, r/o) : Date of the last sync.

OmniFocus 字典

AppleScript 是一门面向对象编程的语言，所以类（和子类）是基本组成元素，OmniFocus 中所常见的上下文、项目和动作等元素，本质上都是相应类的实例（实例化的类通常称为对象）。在 `OmniFocus Suite` 中就可以看到 `folder` 等类。同时类通常会带有一些属性，例如我们之前就将动作的 `name` 属性修改为 `note` 属性。

最后是命令，AppleScript 中的命令可以类比为其他语言的方法，通常可以实现软件内的特定操作。例如 `undo` 命令就可以撤销上一次操作。

字典其实就是应用开发者编写的文档，里面详细介绍各个类、属性、命令的用法，这里不过多赘述。

Apple 官方文档

关于 AppleScript，下列几份 Apple 官方文档值得一看：

- Introduction to AppleScript Overview，基本介绍了 AppleScript 的概念和用处。
- AppleScript Language Guide，最为官方权威的文档。最有价值的部分是其中的 class、commands 和 operators 等参考文档，全面详细地介绍了 AppleScript 的默认关键词的用法。
- AppleScript Release Notes，AppleScript 的更新日志，可以根据版本查看 AppleScript 的改变。

小结

毫无疑问，和 AppleScript 结合起来使用的 OmniFocus，才能真正称得上一款高效率的任务管理工具。

或许你之前对 AppleScript 还心生恐惧，但希望经过本章，你会对 AppleScript 有了基本的认识，甚至能理解它背后的原理。其实对于大部分读者来说，能灵活地使用 AppleScript，就[illegible]能发挥 OmniFocus 最大的价值。当然如果你愿意继续深入探索 AppleScript，会发现 OmniFocus 可以变成任何你想要的样子。

与此同时，AppleScript 也是一座桥梁，使得 OmniFocus 可以和其他工具相结合，形成一套顺手的工作流。这正是我们下一章要探索的内容。

---

1. 最开始是为 Mac OS X 的前身 NeXTSTEP 系统，也就是乔布斯离开苹果创立的 NeXT 公司开发的操作系统。 ↩
2. 详细论证可查看「结构化程序理论」。 ↩

9

上一期
第五章 |「回顾」...

下一期
答疑汇总 05

评论将在经过作者筛选后公开展示

发送

精选评论（3）　　**我的评论**

胡彪

Set A to B，究竟是将谁的值赋给谁呢？为何作者的解释和程序案例的实际用法是相反的呢？

05月30日

sainho 作者

你好，【原始脚本将动作的标题名（`name`）赋值给了变量 `_url`】这句是我的一处笔误，请多见谅。应该是【原始脚本将 `_url` 赋值给了动作的标题名（`name`）】。Applescript 和一般的编程语言赋值语法习惯相反，使用时需要多加注意。

04月29日

渐磨

作者你好，并没有在文章里看到怎么安装AppleScript呀，希望能加入这个内容。

05月30日

sainho 作者

你好，安装 AppleScript 的内容在第一章和第二章中就已经介绍过了。

05月30日

谢浩东Tommy

Apple script很强大,配合hazel或者KM可以实现自动化。收到特定订单文件,自动在of中创建相应的任务或者项目。这一部分也是of学习成本比较高的一部分

02月12日

支持我们 作者招募 用户协议 FAQ Contact Us

© 2013-2018 少数派 | 粤ICP备09128966号-4 | CC BY-NC 4.0